Performance Evaluation of Scheduling Algorithms for Real Time Cloud Computing Systems

Thesis submitted to the department of

Computer Science and Engineering

of

National Institute of Technology Rourkela

in partial fulfillment of the requirements

for the degree of

Master of Technology

by

Varri Murali Krishna

(Roll- 213CS1143)

under the supervision of

Prof. Pabitra Mohan Khilar

Department of Computer Science and Engineering

National Institute of Technology Rourkela

Rourkela- 769008, India

2015

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela - 769008

CERTIFICATE

This is to certify that the thesis entitled **"Performance Evaluation of Scheduling algorithms for Cloud Computing Systems"** submitted by **Varri Murali Krishna**, bearing roll number **213CS1143**,to the Department of Computer Science and Engineering, in partial fulfillment for the award of the degree of **Master of Technology** (**Computer Science**), is a record of bonafide research work carried out by him under my supervision during the period 2014-2015. The thesis has fulfilled all the requirements as per the regulations of the Institute and in my opinion, has reached the standard needed for submission.

Prof. Pabitra Mohan Khilar
Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela-769008, Odisha, INDIA

Acknowledgements

In the first place I would like to record my sincere gratitude to my advisor Prof. Pabitra Mohan Khilar Sir for his invaluable guidance, constant encouragement and mindful attention during the project work giving me extraordinary experiences through out the work. It is indeed an honour and a great privilege for me to have learned from his expertise and experiences which exceptionally inspired and enriched my growth as a student and a researcher.

I owe my deepest gratitude to the entire faculty members of Department of Computer Science and Engineering for providing me an excellent academic environment and support.

I must also convey my heartfelt thanks to the ever diligent staff of Department of Computer Science and Engineering for providing support in everything we did.

This thesis would not be possible without the support of my family members, who have been backing me up throughout my life.

I am indebted to all my classmates for the motivation and support they gave me without any hesitation during the course of my work.

Varri Murali Krishna

May 31, 2015

Abstract

Cloud computing shares data and offers services transparently among its users. With the increase in number of users of cloud the tasks to be scheduled increases. The performance of cloud depends on the task scheduling algorithms used in the scheduling components or brokering components. Scheduling of tasks on cloud computing systems is one of the research problem, Where the matching of machines and completion time of the tasks are considered. Tasks matching of machines problem is that, assume number of active hosts are Y, number of VMs in each host are Z. Maximum number of possible Virtual Machines(VMs) to schedule a single task is (y×z). If we need to schedule X tasks, number of possibilities are $(y \times z)^x$. So scheduling of tasks is NP Hard problem. NP Hard means this scheduling of tasks on VMs not having polynomial time complexity, but it may have algorithm for verifying solution.

Fault-tolerance becomes an important key to establish dependability in cloud computing system. In task scheduling, if task not completed in it's deadline ,then it is one type of fault in scheduling of tasks. In this thesis this type of faults are taken and try to over come it.

In this thesis we present a non-preemptive scheduling algorithm, By inserting the ideal time for postponing the task by ensuring the other task will completes its execution with in the deadline. In simulation the proposed algorithm maximizes the profit of 25%, throughput of 25% and minimizes the penalty of 20% over EDF.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

The computer scientists and mathematicians studied the problem of scheduling for several decades[8][4][3]. The difficulty of scheduling problem is the tasks were preemptive each other or the tasks were not preemptive each other. The data centers are having hosts. Suppose there are 'x' number of tasks,'y' number of hosts and each virtual machine having 'z' number of virtual machines. So these tasks are mapping to the virtual machines in $(y \times z)^x$ ways. So The scheduling of tasks is a NP Hard problem.

The general scheduling problem schedules the tasks according to the various conditions. A task is characterized by ready time,execution time,deadline and requirement of resources. While executing the tasks, the task may be interrupted or may not be. If the task is interrupted then it is known as preemptive scheduling. The tasks having the constraints are based on the precedence relation. The execution of a task will start after completing the execution of its predecessors tasks. The tasks which executed are characterized by the amounts of resources available in the cloud. The goals in the scheduling is improve the utilization, reduce the context switch due to the preemption, and reduce the communication cost.

When the tasks are scheduled on the virtual machines, there may be a chance of occurring the faults. The fault may occur either in the task or in the machine. If the failure occurs in the tasks then the task restart job execution from the check points. If the fault may occur in the machine then the task assigned to that

machine will migrate to another machine. We have presented a lazy evaluation algorithm for executing the tasks. In this approach, the task will wait for some idle period before it starts it's execution.

Task migration can be done in two ways, either the copy of the task may be migrated to another machine or the entire task will migrate to another machine. For the temporary faults there is no need to do migration. The task will execute on the same machine. In our proposed algorithm, if any failure occurs then the fault detection and recovery algorithm is used to avoid the failures.

The characteristics of the cloud computing are the virtualization, distribution and dynamic extendibility. In the present days, software and hardware are provided for supporting the virtualization. Many factors such as IT resource, software, hardware, operating system and net storage are virtualized. The management of the resource in the platform is called as the cloud computing. The dynamic service provider of a cloud uses very large virtualized and scalable resources over the Internet. Cloud computation is defined as the collection of computing and communication resources over the distributed data centres and is shared by many different users. Cloud computing has the most emerging paradigm area in the Information Technology(IT). To judge the quality of a real time application or services the major criteria is time. The real time services over the Internet, all the tasks will meet their deadline guarantee such as hard real time systems.

In the presence of the faults, the fault tolerant computing deals with the computing systems. A fault tolerant system may tolerate to one or more fault types including (i) transient, intermittent or permanent hardware faults, (ii)hardware and software design errors, (iii) operator errors (iv) externally induced upsets or physical damage. In the past years, lots of research has been done on the fault tolerant machines. Most of them are dealing with the hard ware faults. Hardware and software redundancy are well-known effective methods for hardware fault-tolerance, where extra hard ware (e.g., processors, communication links) and software (e.g., tasks, messages) are added into the system to deal with faults. The software fault refers to when the task is not going to complete their execution

with in the deadline. Therefore, in this thesis, I have considered the timing fault. The timing fault of the tasks are recovered by postponing the execution of the tasks, in the cloud[2].

1.2 Real Time Tasks Scheduling : Case Study

Missile Guidance System[2]: In the missile guidance system the computers are placed on the missile. A guided missile has the capable of sensing or tracking the target place. The deviation is calculated by the mounted computer and placed on the missile from the required trajectory and changes the track of the missile to guide it onto the target.

In an **Air Defense System**[2]: Monitoring the incoming enemy missiles in the sky. The below example also highlights important characteristic of real time applications. Since the behaviour of the controlling real time computing system must be predictable, every incoming enemy missile must be destroyed without fail.

Computer On Board an Aircraft[2]: The modern aircraft has the auto pilot option selected by the pilots. After selection the aircraft switches to auto pilot mode then the control of the aircraft is taken by the on board computer. The computer takes the control of take off, navigation, landing the aircraft. The computer checks the acceleration and velocity of the aircraft.

1.3 Literature Review

Table 1.1: Observation on Scheduling Algorithms for Real Time Tasks

Researcher	Year	Observation
Jung, Daeyong, et al.[4]	• 2013	• Proposed VM migration scheme reduces the rollback time and the task waiting time when an instance occur the out-of-bid situation.
Shivam nagapal et al.[7]	• 2013	• Proposes the advantages of the checkpoint scheme is that the checkpoint is made in the end after produces the result of the all nodes. This checkpoint scheme will not cause domino effect.
Cecilia Ekelin et al.[1]	• 2006	• Presented a non preemptive scheduling algorithm called Clairvoyant EDF(CEDF).
R.Santhosh et al.[8]	• 2013	• Presented a online, preemptive scheduling with task migration algorithm for cloud computing environment is proposed in order to improve the efficiency and to minimize the response time of the tasks.
Dilbag singh et al.[10]	• 2012	• Proposed an approach for providing the more availability for the requests of cloud clients.

1.4 Motivation

- The cloud computing system deals with various tasks and resources. The tasks are subjected to various kinds of faults.
- The main fault tolerance issues in cloud computing are detection and recovery. The fault tolerant model is designed in cloud to provide a dependable solution.
- Scheduling of tasks on various resources in a fault tolerant manner has been improved in recent years.

1.5 Objective of The Thesis

1.To design a fault tolerance model which can provide reactive and proactive fault tolerance in real time cloud computing.
2.Maximize the profit and Throughput.
3.Minimize the penalty.

1.6 Thesis Outline

A brief introduction of Real Time Systems ,Real time task scheduling and cloud computing,than case study,than related work ,motivation and problem statement are presented in Chapter 1.
The rest of the thesis organized in to the following chapters :

In chapter2, we discussed 5-3-4-6 principles of cloud computing, Fault tolerant techniques and applications of fault tolerance.

In chapter3, we give the brief description about the task scheduling with fault tolerance by using EDF and Proposed algorithm.

Chapter 2

5-3-4-6 priciples of cloud computing

2.1 Introduction

The aim of the thesis is to schedule the real time tasks on providing performance parameters such as profit, utility, and throughput. We specify the system model for the real time scheduling of tasks that will be used throughout the thesis.

In a chosen time frame physically affected is the purpose of the real time systems.A real time system having two types of systems. The first one is computer called controlling system and another is called controlled system. Based on the availability of information, the controlling system interacts with the environment. The real time system is differentiated from the non real time systems with a common characteristics.

- **Timing constraints:** The timing constraints are impacted on the timing behaviour of the tasks in terms of its release time and absolute time or relative real deadlines of tasks. If a scheduled task will meet its timing constraints then we say that the system works correctly.

- **Safety Criticality:** In the non real time systems the safety and reliability of services are independent issues, but in the many real time systems the safety and reliability are the major issues interactively bounding together to keep it safety.

To maximize the utilization of resources in the cloud computing is the major concern in the Real time systems. So the tasks are mapping to the virtual machines in such way that it improves the overall performance and utilization of the system also increases. In the real time services the tasks complete their execution before

the deadlines such that the system improve in throughput and efficiency. A real time system is generally a controlling system , often embedded into other equipment. It takes in information from its environment, processes it and generates a response.

Cloud Computing[4]: Cloud computing provide the computing resources to the users over the Internet. Instead of storing the data in the hard disk, we store the data in virtual resources available in the cloud. To access the data from the virtual resources we need the Internet. Cloud services provide the software's from third party service providers and users can use this software without installing the software in their local machines. The cloud services include online file storage, web mail, social networking sites and enterprise applications. Cloud computing models can access the data from anywhere in the world. The Architecture of the cloud computing is shown in fig.2.1

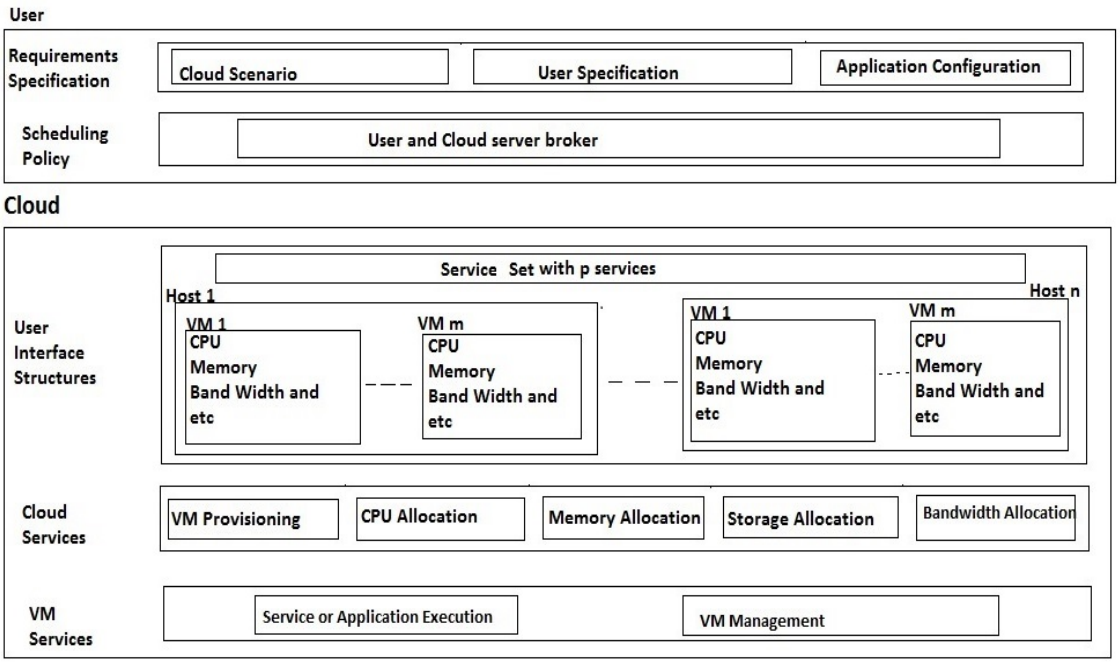

Figure 2.1: Cloud Computing Architecture

The definition of the cloud computing has been developed by the U.S. National Institute of Standards and Technology (NIST):

The cloud computing is a on demand network access to shared a computing re-

sources like storage, servers, network and applications. The computational resources are managed by the very less effort and convenience. The cloud computing models are categorised in to three service models, four deployment models and five important characteristics.

2.2 The Essential Characteristics of Cloud Computing

The five essential characteristics of cloud computing that offers businesses today. They are on-demand self service, broad network access, resource pooling, rapid elasticity, measured service.

On-demand self service: Without human interaction the cloud computing resources are provisioned to the users. This is mostly done though a web based self service portal.The customers can manage their own computing resources are the on-demand self service. Cloud host provider will secure cloud hosting services to the business. We have access to your services and we have the power to change cloud services through an online control panel or directly with the provider. We can change the storage networks ,software as needed and we can add or delete the users. We have to pay the monthly subscription to the service provider and it may vary to each provider.

Broad network access: The resources are accessible over the Internet and supporting heterogeneous client platforms such as mobile devices and workstations in the cloud. The access point is used to get the resources from the cloud. While moving the devices also we can get the services from the cloud. So the users at the top level of the business. This broad network access the private network with in the firewall of the company.

Resource pooling: From the pool of resources the customers can access their own resources. The logical level separates the resources.

Rapid elasticity: To suit our business models the cloud should be flexible and scalable. The resources, users and software's are can be added or removed in the future. At any point of time the application will have the exactly capacity it needs.

Measured service: The mesured service is the customer can pay the bill based on their usage. Your bill based on the measures of the usage, and the number of user accounts. The resources can be monitored from the user side and provider side, based on this the users can pay their bill.

2.3 Service Models

The cloud computing has three service models.

Software as a Service: In The SaaS, The applications and resources are accessed over the Internet by using the web browser. The users can uses the software without installing the software in to their local machines. So the users are not concerned about the operating system,servers,storage,platform, etc. The software as a service is popularized by the salesforce.com, based on the subscription basis which distributes the software rather than the on-premise basis. The SaaS model is used in the business applications including business accounts, collaboration and business management.

Platform as a Service: The customers develops or installs its the operating systems and application software's.The PaaS is the platform as a service provide the platform for building, testing, and creating the applications.In PaaS the provider provides a toolkit to the consumers like different SDK to compile their software and host them on the providers resources. This helps the consumer to have a fair control over the resources provided by the Cloud Provider.

Infrastructure as a Service: The Iaas is a infrastructure as a service provides the hard ware and computing resources. The developers will have an authority to access the resources in the IaaS. In the IaaS, the user can sends programs

and related data to the service provider. The service provider's computer does the computation and returns the results to the user. The cloud infrastructure is flexible, scalable, and virtualized. So the cloud can satisfy the user needs. The examples of the IAAS are IBM Blue Cloud, Amazon EC2.

2.4 Deployment Models

Cloud services are available via a private cloud, community cloud, public cloud or hybrid cloud.

Public Cloud: The services provided by a public cloud are available over the Internet and are owned and managed by a cloud service provider. The examples of the public cloud services like online photo storage, e-mail,socila network sites. Some of the enterprise application services also provided by the public cloud.

Private Cloud: The services provided by a private cloud are the cloud infrastructure is managed or operated by the solely for a specific organization. The services are managed by the organization or a third party service provider.

Community Cloud: The services provided by a community cloud are shared by several organizations and the services are available only to those group of organisations. The organisations or the third party service providers owns and operates the infrastructure.

Hybrid Cloud: The hybrid cloud is a combination of any different methods of resource pooling (for example it may combination of public and community clouds).

Cloud Host[**8**] The large number of systems are integrated and they act as a singli machine in cloud computing technologies. The hosting solutions are depends on the single machine only but the security services are provided by many servers. The advantage of the cloud technology will integrates the resources such as ram or space and that will increase the website improvement.

A **Virtual Machine(VM)**[9] is usually a program or operating system, which does not physically exist but is created within the another environment. A virtual machine has two components : the host and the guest. The host is the virtual machine host server , it process the memory,disk and network I/O and processing power provided by the computing resources.The independent instance of an operating system and application software are separated by the guest. In the host virtual machine has the virtual workloads are called guests.

2.5 Fault Model

The sequence of the system's external state is called the service. If the system is deviated from correct state to fault state because of the one or more external states is called service failure. The failure of the system is caused by an error. The fault is an cause of the error.

Locality is based on the system boundary faults. That can be classified as

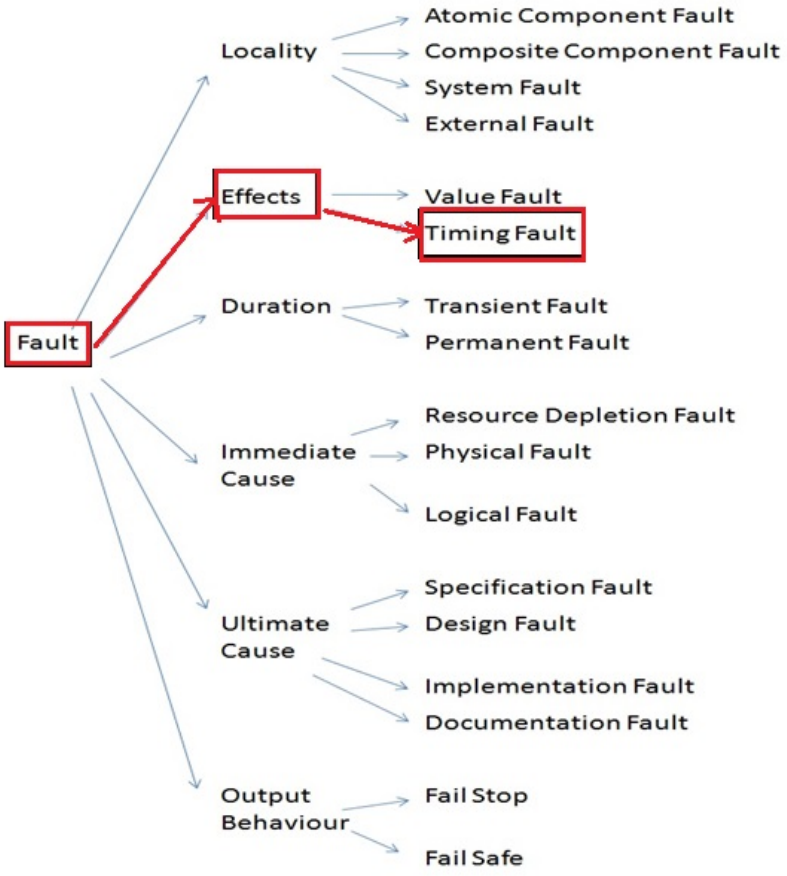

Figure 2.2: Fault Types

- **Atomic component fault:** is the fault in the component cannot be sub-divided.
- **Composite component fault:** is the atomic component faults can be Aggregated in to a composite fault.

- **System fault:** is fault in the structure of the system.
- **External fault:** is the fault caused by the environment or by the users.

Effects does not meet the system's specification because of the deviations.

- **Value fault:** Returns the computation result that does not meet system specification.
- **Timing fault:** The service is not delivered in time.

Duration is the fault persists with respect to the time duration.

- **Transient fault:** Recovers even though a server fails .
- **Permanent fault:** Does not recover because of the server failure.

Immediate happened when the system disobeys the system's specification

- **Resource depletion fault:** To perform the task, system unable to receive the resource.
- **Physical fault:** Hardware breakage or a mutation occurs in executable software.
- **Logical fault:** System does not respond correctly according to the system's specification.

Ultimate happened during the phase cycle of the system development

- **Specification fault:** The requirement specifications are not proper.
- **Design fault:** The requirement and system design are not matched.
- **Implementation fault:** The design implementation and system implementation are not matched.
- **Documentation fault:** The documented system does not belongs to the real system.

Output According to the state of the system behaviour when it produces a fault.

- **Fail-stop:** After reaching the maximum number of faults then the system simply stopping.
- **Fail-safe:** Even though a system leads to failure the Unit gives out certain result. Example is Traffic light controller.

2.6 Fault tolerance Techniques

The cloud computing effected by the various types of faults due to its virtualization and Internet based computing. The fault tolerant techniques can be used in the task level or in the work flow level based on the fault tolerant policies.

2.6.1 Reactive Fault Tolerance

The effect of failures will be reduced by using the policies of the reactive policies of the fault tolerance.

- **Checkpointing:** If a fault occur in the task while executing a task in the virtual machine the task will restarts its execution from the previously state. For developing the big applications, the checkpoint technique is efficient.

- **Replication:** The replication of the task means copy of the task. Different tasks are executed on the different resources at the same time to get the correct result. Replication can be done by using the tools like HA-Proxy, Hadoop and AmazonEc2.

- **Job migration:** If the task has not completed it's execution on a particular machine then the task has to migrate to different machine for completing the task successfully. This can be implemented by using the HA-Proxy.

- **Task resubmission:** When the task is going to fail on a particular machine then the task will resubmit to either the same machine or to a different machine.

- **User defined exception handling:** Based on the failure the user can defines the treatment to remove the faults.

- **Rescue workflow:** The work flow of the system will continues until the system doesn't move forward without tolerate the failed task.

2.6.2 Proactive Fault Tolerance

In this policy we can avoid the faults by predicting them early and replace the suspected components. So that the faults will not come what actually come.

- **Software Rejuvenation:** The system will reboots for a periodic times. So that the system will starts from a clean state.
- **Proactive Fault Tolerance using Self Healing:** It automatically tolerates the faults When multiple instances of an application are running on multiple virtual machines.
- **Proactive Fault Tolerance using Pre-emptive migration:** Preemptive Migration depends on a feedback-loop control mechanism where the application is constantly monitored and analysed.

We injected the fault as a timing fault and tolerate it by using user defined exception handling.

2.7 Applications of Fault Tolerance Computing

Fault tolerant processing are utilized as a part of discriminating real time applications where we require continuous control for instance; power plant, clinics, air ship, weapon, and so on. Long-life applications, for example, unmanned space apparatus need to be profoundly tried and true which can be accomplished by joining issue tolerant strategies into the framework. Essentially, here are high accessibility applications for instance; electronic exchanging framework, OLTP, system exchanging hardware additionally need flaw tolerant. In numerous applications upkeep operations are greatly excessive, awkward, or hard to perform plan the framework so unscheduled support can be maintained a strategic distance from in phone exchanging frameworks, shuttle by adaptation to non-critical failure[2].

2.8 Concepts and Terms

2.8.1 Scheduling and Dispatching

Scheduling is the creation of a schedule: a (partially) ordered list specifying how contending accesses to one or more sequentially reusable resources will be granted. A schedule is intended to be optimal with respect to some criteria (such as timeliness ones).
Number of algorithms proposed for scheduling real time tasks, main classes of algorithms are defined as follows:

- **Preemptive.** The running task will be interrupted at any point of time after assigning the task to the processor for active any other task is called preemptive. This will be according to a predefined scheduling policy.
- **Non-preemptive.**If any task will starts it execution on the proceess, it will execute until the task is completed by the processor.
- **Static.** The scheduling decisions are based on the fixed parameters before submitting the tasks to their execution.
- **Dynamic.** The scheduling decisions are based on the dynamic parameters that may change durin the execution of the tasks. This will improves the resource utilization.
- **Off-line.** The scheduling algorithm executes the entire taska before their actual execution in the off-line. The schedule obtained in this way will stored in to a table and then executed by the dispatcher.
- **On-line.** The scheduling decisions are made at runt ime in the online scheduling algorithm. There may be a new task enters into the system or the running task terminates from execution.
- **Optimal.** An algorithm minimizes the cost function defined over the task set is called optimal algorithm. If there is no cost function defined then there is the only concern to achieve a feasible solution.

- **Heuristic.** An heuristic algorithm provides a feasible solution but it may or may not be a optimal.

In contrast, **dispatching** is the process of granting access to the currently most eligible contending entity.

2.8.2 Schedulable and Non-Schedulable Entities

Schedulable entities (e.g., threads, tasks, and processes in both the application and the system software) are scheduled by the scheduler. Non-schedulable entities are most often in the system software and can include interrupt handlers, operating system commands, packet-level network communication services, and the operating systems scheduler. Non-schedulable entities can execute continuously, periodically, or in response to events; their timeliness is a system design and implementation responsibility. Now, we present the following definitions for the scheduling which are available in the literate.

Definition 1 ***Valid Schedule*** *A valid schedule of a set of tasks is a schedule which satisfying the following properties[10]*

- *Each process can only start execution after its release time.*
- *All the precedence and resource usage constraints are satised.*
- *The total amount of processor time assigned to each task is equal to its maximum or actual execution time.*

Definition 2 ***Feasible schedule*** *A feasible schedule of a set of tasks ψ is a valid schedule by which every task completes by its deadline a set of tasks is schedulable according to a scheduling algorithm if the scheduler always produces a feasible schedule[10].*

Definition 3 ***Optimal schedule*** *An optimal schedule of a set of tasks ψ is valid schedule of ψ with minimal lateness. A hard real-time scheduling algorithm is optimal if the algorithm always produces a feasible schedule for a given set of tasks[10].*

2.8.3 Timeliness Specification

In real time systems based on timeliness specification real time tasks can be of the following type[1].

1. **Deadline:** A deadline is the time, in real time systems, task are acceptable only if it complete it's execution before the dead line. If task not complete it's execution then it will be rejected. Some times this rejection can causes more loss. In real time systems task may complete it's execution before deadline or may not. Some time not complete in deadline is also allowed, based on that real time systems are soft and hard[6].

2. **Hard Deadline** A hard real time system is one in which a failure of the system to meet the specified worst case response time to an input or real world event will lead to overall system failure. Hard real time systems are easy to spot from their specifications which talk about maximum response times and the effects of failure to meet that time. For example, a system specification might state that given input X the system will respond with output Y within 10ms. If it takes 10.1ms then the system could well fail and you are out of a job[6].

   Example Hard Real Time Systems: Heart Monitoring System used in Coronary Care. Engine Management, ABS, Stability and Traction Control systems in a car. Nuclear Power Station controlling the reaction. Air Traffic Control system or Auto-pilot in an aircraft or helicopter. Digital Signal Processor such as CD/MP3 player or digital filter (time is especially critical here if acceptable sound is to be produced).

   In other words, failure of a hard real-time system often results in complete failure of the system, sometimes incurring significant cost and loss of life or serious injury and damage. In summary, hard real time systems are time critical.

3. **Soft Deadline** A soft real time system implies that failure to meet a specified response time merely results in system degradation not necessarily

outright failure. A Soft Real time systems specification will thus quote a typical, suggested or average response time against which degradation (not necessarily failure) can be judged. The response time of a Soft Real Time system is thus not fixed and may improve or degrade within acceptable limits depending upon system loading. Of course system degradation can ultimately become system failure if the response time becomes so intolerable that it no longer functions acceptably[2].

Example Soft Real Time Systems *An Elevator control system* Response time varies with the time of day and passenger load. This could be said to have failed if all the passengers decide it is quicker to walk instead, but we might consider it to work acceptably if 95% of passenger requests are met within 30 seconds, averaged over a day.*An ATM for a bank* Response time varies with the time of day and load experienced by the bank central computer. Failure could be said to have occurred if the transaction cannot be completed without annoying the customer (say 2 mins). However success might be measured by a system that on average completes 99.5% of transactions within 1 min.

2.8.4 Task Scheduling

The most important thing in RTS is meeting task deadlines as explained above. Scheduling of tasks involves the allocation of processors(including resources) and time to tasks in such a way that certain performance requirements are met[5]. The task scheduling depends on the system performs schedulability and the task are executed whether in the statically or in the dynamically and the result of the tasks dispatched at runtime.

2.9 Conclusion

In this chapter, we discussed the essential characteristics of the cloud,service models, deployment models, fault model fault tolerate techniques,application of fault tolerance computing and scheduling and dispatching in the scheduling of real time tasks.

CHAPTER 3

Algorithm for Fault Tolerant Real Time Scheduling in Cloud

3.1 Introduction

For a given task set, the problem is to generate a feasible solution. In the EDF algorithm all the tasks are scheduled based on the their dead line times. All the tasks are needed to complete their execution before the deadline and the task will not start early before its ready queue arrival time. Lets assume the online scheduling for real time system so the time is crucial.

The EDF is non-optimal for non-preemptive systems, The other algorithms not guarantee to schedule more tasks than EDF. The future task arrivals are not considered in the in the EDF. Let us assume the arrival times of the future tasks are known. So in a non-preemptive system it is some time necessary not to dispatch the tasks in the ready queue if it will prevent to meet the deadlines of future tasks. For not dispatching the tasks in the ready queue we insert the idle time. Due to the NP-completeness of the scheduling problem the problem is when to insert idle time for any algorithm. Table 3.1 provide various for the proposed task scheduling algorithm for cloud.

SI	Notation	Description
1	H	Host set.
2	H_a	Active Host set.
3	h_i	i^{th} Host.
4	V_i	i^{th} Host VM set.
5	v_{ij}	i^{th} Host j^{th} VM.
6	T	Task set.
7	t_k	k^{th} task in set T.
8	a_k	k^{th} task arrival time.
9	e_k	k^{th} task execution time.
11	d_k	k^{th} task deadline.
12	st_k^{min}	possible earliest starting time of the task t_k.
13	st_k^{max}	The possible latest start time of the task t_k.

Table 3.1: Tasks Scheduling Parameters

3.2 Problem Description and Task Scheduling Model

$\Longrightarrow$ Let a cloud computing system consists of an infinite set of host machines such as $H = \{h_1, h_2, h_3, ...\}$.

$\Longrightarrow$ Let $H_a = \{h_1, h_2, h_3, ..., h_n\}$; are set of active host machines.

$\Longrightarrow$ Let H be the total host set.

$\Longrightarrow$ Let $V_i = \{v_{i1}, v_{i2}, v_{i3}, ..., v_{im}\}; m = |V_i|$,are set of virtual machines with in the host h_i.

$\Longrightarrow$ Let us consider a task set $T = \{t_1, t_2, t_3, ...\}$ of independent tasks that arrive dynamically.

$\Longrightarrow$ The task t_k having parameters as $t_k = \{a_k, e_k, d_k\}$ Where, a_k is arrival time, e_k is execution time of task and d_k is deadline time.

In our proposed algorithm some of the special parameters are used they are st_k^{min} is the possible earliest starting time of the task t_k, st_k^{max} is The possible latest start time of the task t_k. initially the earliest starting time of the task st_k^{min}=a_k and the latest start time is st_k^{max}=$d_k - e_k$ with respect to the deadline d_k and the execution timee_k , these properties are not constant. The task depending on the properties of the other tasks and the scheduling decisions. The task must be starts its execution between the interval [st_k^{min} , st_k^{max}]. The interval of the task cannot be increased but it may decreased based on the results of the other task. If the task t_k will postponed then the earliest starting time of the task st_k^{min} parameter would be increased. So by increasing the starting time of this task, another task cannot start as late, st_l^{max}would be decreased.

Check Postpone conditions: The postpone condition is used to decide whether to insert the idle time or not. The first task in the ready queue is represented by t_k, and the first task in the critical queue is represented by t_l. The task in the ready queue must be postponed to later by satisfying the following three conditions.

1. $(st_k^{min} + e_k) > st_l^{max}$
2. $t_k \neq \mathrm{t}_l$
3. $st_l^{min} \leq \mathrm{st}_l^{max}$

The condition 1 says that, executing the task $t_k$earlier than the task t_l. Then the execution of the task t_l is not possible.

The condition 2 says that the task in the ready queue and the task in the ready queue were two different tasks.

The condition 3 says that the task in the critical queue t_l has not misses its deadline before.

If any of the above conditions fails then the task in the ready queue will dispatch to the corresponding virtual machine. The task in the ready queue will postponed to later by satisfying the above conditions.

The postponing of a task, it means $st_k^{min} = st_l^{min} + e_l$ updated the earliest starting time of the task. The postponed task will comes for execution by using

the trigger. This procedure will repeat for the every first task in the critical queue by holding the first condition. In the worst case the critical queue required n updations.

Objective of this thesis is maximizes Absolute Profit, minimizes the Absolute Penalty, and maximizes through of scheduling.

1. **Absolute Profit:**
   Absolute profit is defined as sum of profit of all tasks completed in deadline. In our proposed algorithm will get more profit compared to the EDF. Our objective is to maximize the Absolute profit.
   Absolute Profit = $\Sigma\ t_k$.profit $\forall\ st_k^{min} + e_k \leq d_k$

2. **Absolute Penalty:**
   Absolute penalty is defined as sum of penalty of all tasks misses their deadline. In our proposed algorithm attains less penalty compared to the EDF. Our objective is to minimize the Absolute penalty.
   Absolute Penalty = $\Sigma\ t_k$.penalty $\forall\ st_k^{min} + e_k > d_k$

3. **Throughput:**
   Throughput calculates the number of tasks completed from the arrived set of tasks in the ready queue. So we should maximize this this throughput.
   Throughput = count(tasks) $\in\ st_k^{min} + e_k \leq d_k$

3.3 Scheduling Algorithms

3.3.1 Algorithms Pseudo Code

This section having discussion of the two algorithms EDF, and proposed algorithm

Here Inputs are T:Set of n Tasks, H:Set of m active Hosts, V_i:Set of VMs on Host H_i

Output is Executed Task Set $ETS_{i,j}$ Having tasks, which are successfully meet their deadlines on VM j on Host i

Algorithm 1 EDF

Input: T, H, V_i:Set of VMs on Host H_i

Output: ETS:Successfully Executed Task set

1: MAX=max$\{d_k, \forall t_k \epsilon T\}$
2: **for** $i \leftarrow$1 to MAX **do**
3: RQ is empty set //Ready Queue
4: update RQ with new arrive task, and not assign task which is already arrived
5: Sort RQ as Ascending order of Dead line of Tasks
6: **if** Task on k^{th} host, l^{th} VM complete execution **then**
7: ETS=$\{$ETS$\}$ $\bigcup$ $\{$successful completed task$\}$
8: Free $VM_{k,l}$
9: **end if**
10: **if** Task on k^{th} host, l^{th} VM reach dead line **then**
11: Free $VM_{k,l}$
12: **end if**
13: **for** $m \leftarrow$ each task in RQ **do**
14: **if** k^{th} host, l^{th} VM is free **then**
15: Assign Task RQ_m to k^{th} host, l^{th} VM
16: **end if**
17: **end for**
18: **end for**

In algorithm3.3.1.1 it the scheduling of algorithm in EDF algorithm, which is the existing one. In EDF algorithm the tasks are schedule based on the dead line. In arrived tasks, the tasks which is less dealine time is schedule first. For that sorting of ready queue is done in the ascending order. After that check for availability of the free VM than task with nearest deadline is assign to VM.

Algorithm 2 Proposed Algorithm

Input: T, H, V_i:Set of VMs on Host H_i

Output: ETS:Successfully Executed Task set

MAX=max$\{d_k, \forall t_k \epsilon T\}$
Critical Queue CQ is formed based on ascending order of task's st_k^{max} values
for $i \leftarrow$1 to MAX **do**
    RQ is empty set //Ready Queue
    update RQ with new arrive task, and not assign task which is already arrived
    Sort RQ as Ascending order of Dead line of Tasks
    if Task on k^{th} host, l^{th} VM complete execution **then**
        ETS={ETS} $\bigcup$ {successful completed task}
        Free $VM_{k,l}$
    end if
    if Task on k^{th} host, l^{th} VM reach dead line **then**
        Free $VM_{k,l}$
    end if
    t $\leftarrow$ first task in CQ
    for $m \leftarrow$ each task in RQ **do**
        if $st_m^{min} + e_m > st_t^{min}$ and m $\neq$ t and $st_t^{min} \leq st_t^{max}$ **then**
            $st_m^{min} \leftarrow st_t^{min} + e_t$
        else
            remove m task from CQ and update CQ
            if k^{th} host, l^{th} VM is free **then**
                Assign Task RQ_m to k^{th} host, l^{th} VM
            end if
        end if
    end for
end for

The algorithm3.3.1.2 will maintain two queues namely ready queue and critical queue. From step 14 to step 21 refers the code above incorporated over EDF algorithm in order to improve the performance. In EDF it wont care about the future tasks. If the task will completes its execution has to be removed from the ready queue and critical queue. If the postpone conditions are not satisfied then the task in the ready queue will be dispatched for execution. Task will be postponed to future time by satisfying the postpone conditions.

3.3.2 Time complexity of Two Algorithms

Assume d_{max} be the maximum dead line of all dead lines of task set T,n be the number of active available host,m be the maximum possible VMs in any host,finally p be the maximum number of tasks that can be present in Ready Queue in time span between 1 to d_{max}.

Now time complexity of algorithm is $d_{max} \times (p \log(p) + p \times n \times m + n \times m)$

where d_{max} is the time span of scheduling, it is multiply by three parts, in that $p \log(p) + p$ is multiplied because sorting of ready queue ,here maximum possible task at any time is p, multiplication with $p \times n \times m$ is done because assign of tasks in ready queue to host's VM, multiplication with $n \times m$ is done because free the VMs in each host after complete the execution of task.

Now finally we get

$d_{max} \times (p \log(p) + p \times n \times m + n \times m) = O(d_{max} \times p \times n \times m)$

So time complexity of the two algorithms is $O(d_{max} \times p \times n \times m)$

3.4 Simulation Results

In this section, we present the scheduling model simulation and results.

3.4.1 Scheduling Model

For simulating our proposed algorithms, MATLAB R2009a is used over scheduling model diagram is shown below.

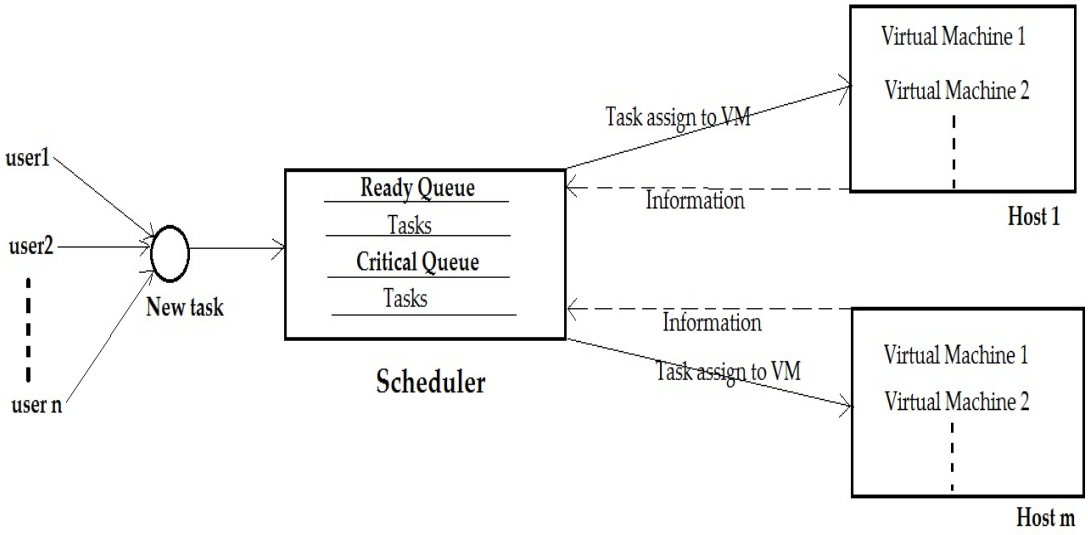

Figure 3.1: Scheduling model

In fig3.1, it is shown the scheduling model of simulation. The scheduling model having two type of queues ready queue and critical queue. It is having scheduler which make decisions centrally as which host, which VM is assign to task in ready queue. The assign task in migrated to host and VM. The task executed successfully on VM than it will be accepted else task will be rejected.

3.4.2 Simulation Assumptions and Performance Parameters

Assumptions of Task, Host ,VM parameters in the simulation model

Task

1. Number of Task in sequence {2,4,6,...,40}.
2. Arrival Time of Tasks in Range(CC) [1,25].
3. Execution time of Tasks in Range (1,75].

4. Dead line of Tasks in Range(CC) (1,125].
5. All are Randomly Generated in mention Range.

Host and VMs

1. Number of Hosts are 2.
2. Number of VMs in each Host 3.
4. All are Randomly Generated in mention Range.

Performance parameters used in this simulation are

1. Absolute Profit.
2. Absolute Penalty.
3. Throughput.

Here Absolute Profit, Absolute Penalty, Throughput is calculated by varying number of task by taking Host and VMs as fixed in number.

3.4.3 Simulation Results

In Simulation Results Graph is drawn between sets {Absolute Profit, Absolute Penalty, Throughput}, {Number of Tasks}, so totally three graphs are shown. After that explanation of that graphs is done.

The fig3.2 is plotted between Absolute Profit and Number of Tasks. Absolute profit is defined as the sum of profits of all tasks completes their execution in deadline. As the number of tasks increasing the number of tasks misses their deadline in the EDF. But in our proposed algorithm less number of tasks misses their deadline compared to the EDF. The result shows that our proposed algorithm maximizes the profit of 25% over the EDF.

To depict the penalty, Fig3.3 shows the variation between the number of tasks and the Absolute penalty. Absolute penalty is the number of tasks misses their deadline. The result shows that our proposed algorithm minimizes the penalty of 20% over the EDF.

The fig3.4 is plotted between Throughput and Number of Tasks. Throughput calculates the number of tasks completed from the arrived set of tasks in the ready

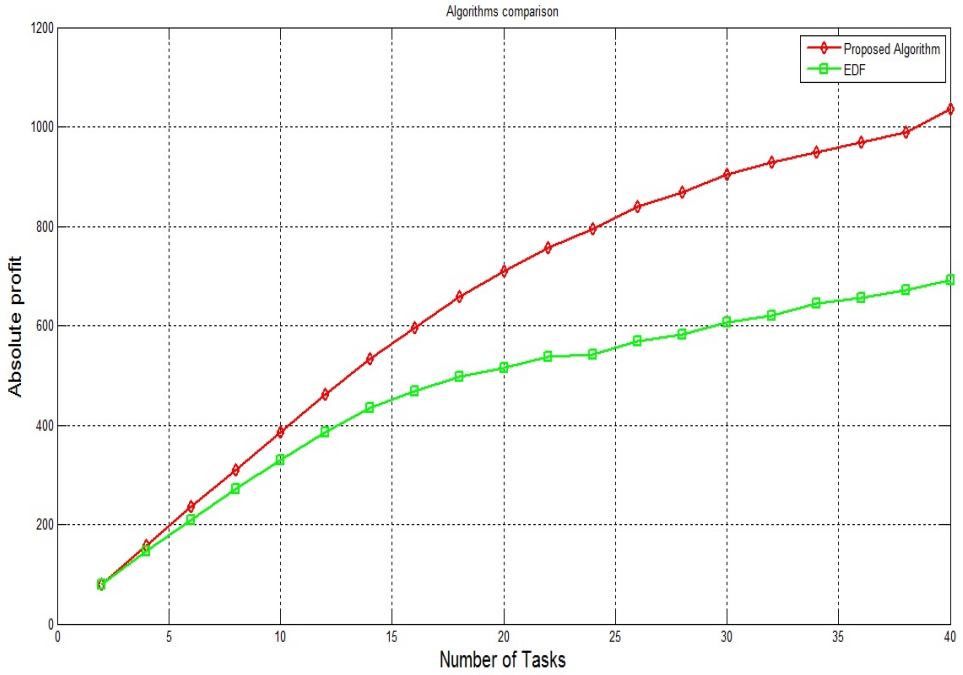

Figure 3.2: Absolute Profit Vs Number of Tasks

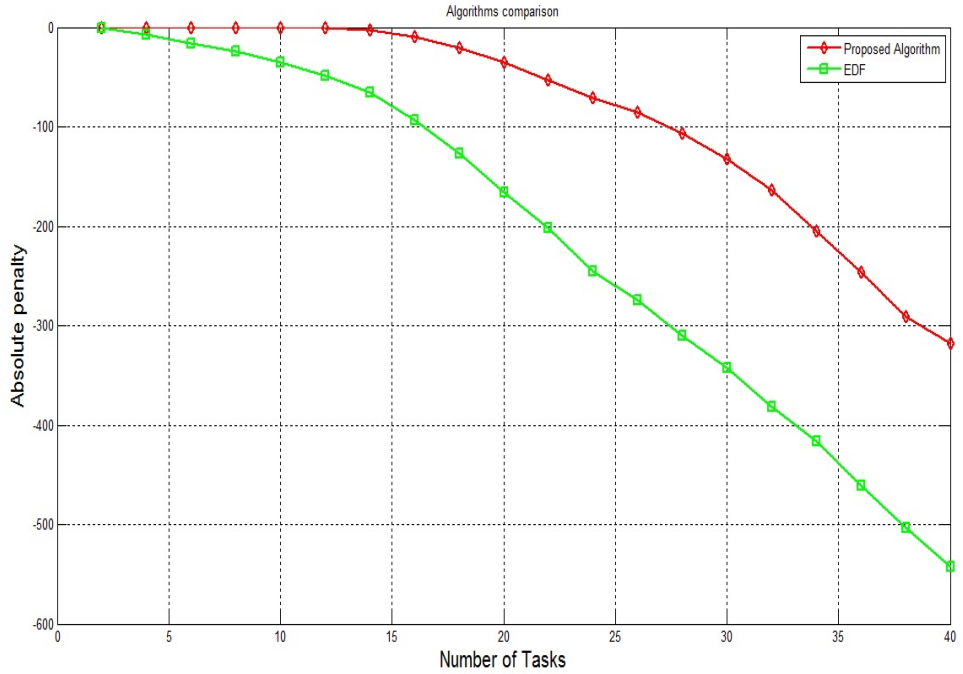

Figure 3.3: Absolute Penalty Vs Number of Tasks

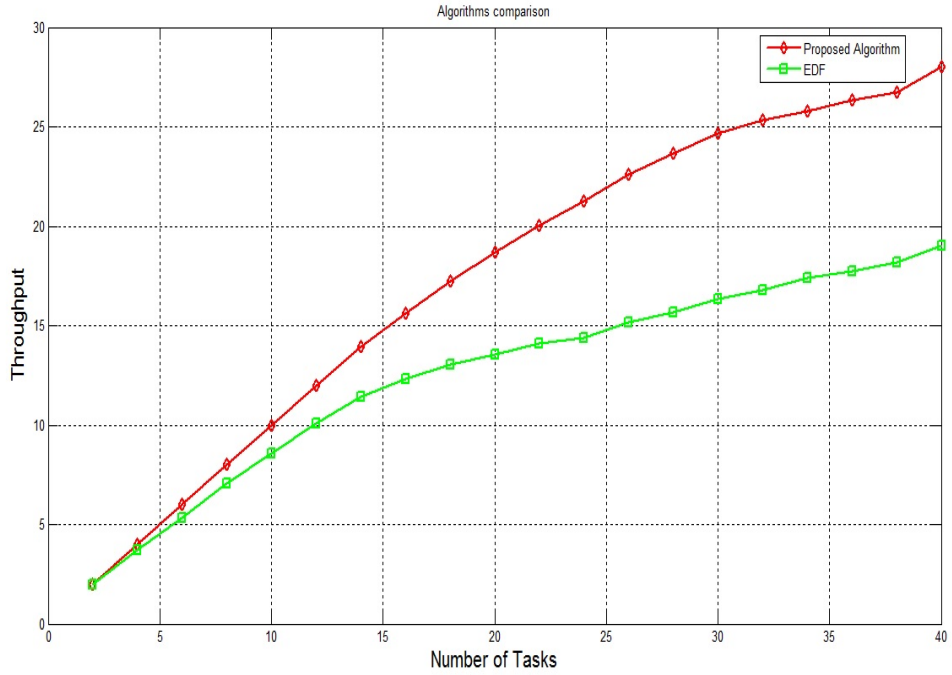

Figure 3.4: Throughput Vs Number of Tasks

queue. The result shows that our proposed algorithm maximizes the throughput of 25% over the EDF.

3.5 Conclusion

In this chapter we have seen that task are scheduled by EDF and proposed algorithm. By simulation results it is shown that the proposed algorithm maximizes the profit and throughput of 25% over EDF algorithm. We have designed a task scheduling model In this chapter, which is simulated for real time cloud computing systems.

Chapter 4

Thesis Conclusion and Future Work

The scheduling algorithms for cloud computing system is an challenging problem in recent years. Fault tolerant scheduling algorithm is needed in order to provide the dependable solution for the cloud. The completion of task in the real time cloud with in the deadline is concluded in my thesis. Although , a cloud may suffer from various kinds of faults, we have provided the solution for timed value fault. Our proposed algorithm maximizes the profit of 25%, minimizes the penalty of 20%, maximizes the throughput of 25% over the EDF.

In the future work we will specify a scheduling algorithm by using the fault tolerance techniques like checkpoint and VM migration. The fault may occur either in the task or in the machine. If the task is faulty then it will starts it's execution from the checkpoint not from the beginning. If the machine is faulty either the copy of the task or the original task will migrate to another machine. By using these fault tolerance techniques we can avoid the faults in the future.

Bibliography

[1] Ekelin, C., "Clairvoyant non-preemptive edf scheduling," in *Real-Time Systems, 2006. 18th Euromicro Conference on*, pp. 7–pp, IEEE, 2006.

[2] Ekka, A. A., *Fault Tolerant Real Time Dynamic Scheduling Algorithm For Heterogeneous Distributed System*. PhD thesis, 2007.

[3] Gao, Y., Gupta, S. K., Wang, Y., and Pedram, M., "An energy-aware fault tolerant scheduling framework for soft error resilient cloud computing systems," in *Design, Automation and Test in Europe Conference and Exhibition (DATE), 2014*, pp. 1–6, IEEE, 2014.

[4] Jung, D., Chin, S., Chung, K. S., and Yu, H., "Vm migration for fault tolerance in spot instance based cloud computing," in *Grid and Pervasive Computing*, pp. 142–151, Springer, 2013.

[5] Malik, S. and Huet, F., "Adaptive fault tolerance in real time cloud computing," in *Services (SERVICES), 2011 IEEE World Congress on*, pp. 280–287, IEEE, 2011.

[6] Menychtas, A. and Konstanteli, K. G., "Fault detection and recovery mechanisms and techniques for service oriented infrastructures," *IGI Global*, 2012.

[7] Nagpal, S. and Kumar, P., "A study on adaptive fault tolerance in real time cloud computing," *International Journal of Advanced Research in Computer Science and Software Engineering (IJarcsse), ISSN*, vol. 2277.

[8] Santhosh, R. and Ravichandran, T., "Pre-emptive scheduling of online real time services with task migration for cloud computing," in *Pattern*

Recognition, Informatics and Mobile Engineering (PRIME), 2013 International Conference on, pp. 271–276, IEEE, 2013.

[9] Sharma, R. and others, "Task migration with edf-rm scheduling algorithms in distributed system," in *Advances in Computing and Communications (ICACC), 2012 International Conference on*, pp. 182–185, IEEE, 2012.

[10] Singh, D., Singh, J., and Chhabra, A., "High availability of clouds: Failover strategies for cloud computing using integrated checkpointing algorithms," in *Communication Systems and Network Technologies (CSNT), 2012 International Conference on*, pp. 698–703, IEEE, 2012.